ALGORITMOS Y ESTRUCTURAS DE DATOS III - 2º Parcial
Fecha examen: 07-JUL-2017 / Fecha notas: 12-JUL-2017

	Nº Orden	Apellido y nombre	L.U.	Cant. hojas[1]
Completar:	6			8
	Nota (Nº)	Nota (Letras)	Docente	
No completar:	7,80	siete con 80/100	—	

1. Sean los siguientes problemas.

   Π_1: CIRCUITO EULERIANO
   Entrada: grafo G.
   Pregunta: ¿existe un circuito que pasa exactamente una vez por cada eje de G?

   Π_2: CIRCUITO HAMILTONIANO
   Entrada: grafo G.
   Pregunta: ¿existe un circuito que pasa exactamente una vez por cada vértice de G?

   Considerar una función f que recibe como entrada un grafo y determina en tiempo polinomial si tal grafo tiene un circuito que pasa exactamente una vez por cada eje; en caso afirmativo la función devuelve K_3, y caso contrario devuelve K_3^c. Decidir si f es una reducción polinomial de Π_1 a Π_2. En caso afirmativo demostrar; en caso negativo justificar.

2. Dados dos grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$, un homomorfismo de G_1 a G_2 es una función $f : V_1 \to V_2$ tal que $(v, w) \in E_1 \Rightarrow (f(v), f(w)) \in E_2$. Decimos que G_1 es homomorfo a G_2 si y sólo si existe un homomorfismo de G_1 a G_2.

   Sea G un grafo. Demostrar que G es 2-coloreable si y sólo si es homomorfo a K_2.

3. Sea G un grafo de n vértices y m ejes. Diseñar un algoritmo de complejidad $O(n^2)$ que encuentre un subgrafo completo de G que sea maximal (no necesariamente máximo). Mostrar que el algoritmo propuesto es correcto y determinar su complejidad. Justificar. El mejor algoritmo que conocemos tiene complejidad $O(m + n)$, lo cual es necesario para obtener puntaje máximo en este ejercicio.

4. Sea S el conjunto de números reales no negativos de la forma $a\pi + b$, con $a, b \in \mathbb{Z}$. Sea G una red de flujo. Demostrar que si para todo eje su capacidad pertenece a S, entonces...

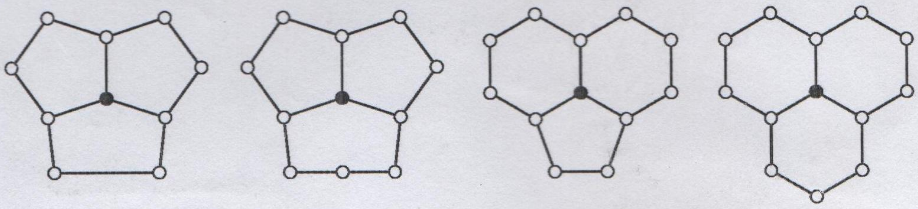
   (a) para todo corte su capacidad pertenece a S. 0.

   (b) para todo flujo máximo su valor pertenece a S. 0.

   (c) existe un flujo máximo que asigna a cada eje un valor que pertenece a S. 0.

   ¿Valen las recíprocas? En caso afirmativo demostrar; en caso negativo dar un contraejemplo y justificar.

5. El grafo de una pelota de fútbol generalizada es un grafo conexo que admite una representación planar donde cada vértice está en el borde de exactamente tres regiones, cada una de las cuales puede ser pentagonal o hexagonal (una región pentagonal es aquella cuyo borde es un ciclo simple de 5 vértices; análogamente para hexagonal). En la siguiente figura se muestran las cuatro posibles situaciones en las que puede estar cada vértice del grafo.



   Sean n y m las cantidades de vértices y ejes del grafo, respectivamente. Sea r la cantidad total de regiones en la representación planar mencionada, de las cuales r_5 son pentagonales y r_6 son hexagonales.

   (a) Demostrar que $m = (5r_5 + 6r_6)/2$. 0.

   (b) Demostrar que $n = (5r_5 + 6r_6)/3$. 0.

   (c) Determinar el valor exacto de r_5. Justificar. 0.

   (d) ¿Es posible que todas las regiones sean pentagonales? En caso afirmativo exhibir un grafo que lo cumpla y justificar; en caso negativo demostrar. 0.2

   (e) ¿Es posible que todas las regiones sean hexagonales? En caso afirmativo exhibir un grafo que lo cumpla y justificar; en caso negativo demostrar. 0.2

[1] Incluyendo a esta hoja. Entregar esta hoja junto al examen.

H: 1/8

Ejercicio 1:

NOTA: 2

Si lo es, esto porque para f ser una reducción polinomial de Π_1 a Π_2, f tiene que cumplir:

- transformar instancias de Π_1 a instancias de Π_2
- ser polinomial
- y cumplir que sea $I \in D_{\Pi_1} \Rightarrow I \in Y_{\Pi_1} \Leftrightarrow f(I) \in Y_{\Pi_2}$

• transformar instancias:

f toma un grafo como entrada, que justamente son las entradas de Π_1. Y devuelve como resultado a K_3 o K_3^c que en particular son grafos, y como Π_2 no tiene restricciones sobre la entrada es válida.

• f es polinomial:

para representar a los nodos vamos a utilizar matriz de adyacencia. Por lo que el tamaño de la entrada va a ser de t^2.

Luego como vimos en la teórica calcular ~~[illegible]~~ si un grafo tiene un circuito ~~[illegible]~~ euleriano es un algoritmo polinomial en cuanto al tamaño de la entrada (chequear que todos los nodos tenga grado par). Y CONEXO + NODOS AISLADOS

Y luego una vez obtenido el resultado crear un K_3 o un K_3^c que es una operación de $O(1)$ ya que siempre está acotada por $t \cdot 3^2$ ~~que es lo que~~

Ya que 3^2 siempre es lo que cuesta crear una matriz de 3 nodos, con representación de matriz de adyacencia. ✓

- Ver que si $I \in Y_{\pi_1} \Leftrightarrow f(I) \in Y_{\pi_2}$

$\Rightarrow$) Si ~~$\not I$~~ $I \in Y_{\pi_1}$

$\Rightarrow$ I tiene un circuito euleriano

$\Rightarrow$ $f(I)$ devuelve K_3

$\Rightarrow$ K_3 tiene un ~~camino~~ circuito Hamiltoniano

$\Rightarrow$ $K_3 \in Y_{\pi_2}$ ✓

$\Leftarrow$)- Si $f(I) \in Y_{\pi_2}$

$\Rightarrow$ $f(I)$ tiene un circuito Hamiltoniano

$\Rightarrow$ $f(I)$ es igual a K_3. Porque si ~~es~~ fuese igual a K_3^c no ~~[illegible]~~ se cumpliría que tiene un circuito Hamiltoniano, ~~[illegible]~~ ✓

$\Rightarrow$ I tiene un circuito euleriano.

K_3 tiene circuito Hamiltoniano

CLARÍSIMO

Sea $K_3 = (\{v_1, v_2, v_3\}, \{(v_1, v_2), (v_2, v_3), (v_3, v_1)\})$

el circuito es $\langle (v_1, v_2), (v_2, v_3), (v_3, v_1) \rangle$.

K_3^c no tiene un circuito Hamiltoniano:

K_3^c tiene más de una componente conexa, lo que hace absurdo que tenga un circuito que pase por todos los nodos.

Ejercicio 2) - Sea $K_2 = v_1 \text{———} v_2$ y $E_2 = \{(v_1, v_2)\}$

2p

Podemos decir en otras palabras que

G es homeomorfo a K si $\exists f$ tq

$f : V_1 \longrightarrow \{v_1, v_2\}$ tal que $\forall (u_1, u_2) \in E_1$

$$(f(u_1) = v_1 \wedge f(u_2) = v_2) \vee (f(u_2) = v_1 \wedge f(u_1) = v_2)$$

$\Leftarrow$) qvq G es 2-coloreable si es homomorfo a K_2.

Para ello vamos a ver que ser homomorfo a K_2 implica que no tiene ciclos impares.

Asumamos lo contrario, G es homomorfo a K_2 y tiene ciclos impares, entonces G contiene un subgrafo C_{2k+1}, de la pinta $C_{2k+1} = \langle u_1, \dots, u_{2k+1}, u_1 \rangle$.

Tenemos que si $f(u_i) = v_1 \iff f(u_{i+1}) = v_2$. Ya que si no el eje (u_i, u_{i+1}) no cumple la condición del homomorfismo (los ejes (v_1, v_1) y (v_2, v_2) no pertenecen a E_2).

Por esto tenemos que si $f(u_1) = v_1 \Rightarrow f(u_2) = v_2 \Rightarrow$ $f(u_3) = v_1 \dots$ y así podemos deducir que, si $f(u_1) = v_1$ entonces $\forall u_i, u_{i+1}$ tq $i \equiv 1$ módulo $2 \Rightarrow f(u_i) = v_1$ y $f(u_{i+1}) = v_2$

$\Rightarrow$ Por esto tenemos que $f(u_{\underbrace{2k+1}_{\text{impar}}}) = v_1$. Pero esto lleva a que la arista $(u_{2k+1}, u_1) = (v_1, v_1)$ que no pertenece a E_2.

lo que genera un absurdo ya que G ~~es~~ es homomorfo a K_2. (Caso donde $f(u_1)=v_2$ es análogo...)

Ya entonces probado que si G es homomorfo a K entonces G no tiene ciclos impares. Por teórica, tenemos que G es bipartito, lo cual es también por teórica es 2-coloreable.

$\Rightarrow$) - Tenemos que G es 2-coloreable entonces qvq es homomorfo a K_2.

Que G sea 2 coloreable significa que se puede colorear con al menos 2 colores, en otras palabras, G se puede separar en 2 conjuntos independientes. donde cada conjunto representa un color. (Conjuntos X_1 y $X_2 \subseteq V_1$). Esto es sinónimo de que G es bipartito, simplemente tomando X_1 y X_2 se logra la bipartición. Por lo que si nombramos a f igual a $f(u)=\begin{cases} v_1 & \text{si } u \in X_1 \\ v_2 & \text{c.c} \end{cases}$. Cumple la condición de isomorfismo, ~~...~~ si porque no va a existir arista (u_i, u_j) tq $(f(u_i), f(u_j))$ sea igual a (v_1, v_1) o (v_2, v_2), ya que si $f(u_i)$ es igual a $f(u_j)$ significa que pertenecían al mismo conj independiente. y por definición del mismo, la arista (u_i, u_j) no existe en E_1

Ejercicio 3)-

Asumo representación del grafo con matriz de adyacencia, para poder obtener si 2 nodos son vecinos en O(1).

```
void GrafoMaximal(Grafo g, vector<int>& res, int n){
    for(int i=1; i<n+1; i++){
        int m = res.size();                          O(1)
        bool esVecinoDeTodos = true;                 O(1)
        for(int j=0; j<tamSol; j++){                 O(1)
            if(!(sonVecinos(i, res[j]))){            O(1)
                esVecinoDeTodos = false;             O(1)
            }
        } ---------------------------------------> O(|tamSol|)
        if(esVecinoDeTodos){
            res.push_back(i); ---------------------> O(|1|)
        }
    } ---------------------------------------------> O(n * |tamSol|)
}
```

(por todos los nodos) y haciendo por cada ite

una ~~operación~~ ciclo que repite O(tamrespuesta)

~~las~~ ejecutando una ~~est~~ operación de O(1) en cada

Y como la respuesta a lo sumo tiene n elementos

esta se puede acotar por n justamente y

queda que el ciclo principal iteró n

veces, con una operación de n en medio.

Eso nos deja una complejidad total de

$O(n^2)$.

El algoritmo es correcto, ~~ya~~ ya que

en primer lugar por como agregamos los nodos a la

solución, ~~estos~~ estos solo los agregamos si

pasa que son vecinos de todos los de la misma.

Por lo que se cumple que la solución resulta

represente un subgrafo completo de G.

Ahora queda ver que no se puede agregar

ningún otro nodo a la solución. Y esto es porque

si se pudiese eso significaría que es vecino de

todos los nodos pertenecientes al vector respu

Por ende en su iteración este hubiese sido agregado. 4: 4/8

Nö: 6

Si se pudiese saber la cardinalidad del conjunto de vecinos en $O(1)$ de un nodo. El algoritmo se podría ~~bag~~ bajar a complejidad $O(n+m)$. Ya que se puede agregar un if-then-else al comienzo de cada iteración del ciclo principal diciendo que si su cantidad de vecinos es menor que el conjunto solución Actual, que esa iteración no opere. Ya que sabemos que dicho nodo no va a ser vecino de todos los de ~~na~~ la solución.

Luego solo haríamos la operación de chequear en los casos que los vecinos tengan un conjunto de vecinos mayor a la sol actual pudiendo acotar esta por ~~mm~~ total vecinos de n_i en la iteración i por lo que queda que la complejidad es de

$$\left(\sum_{\min(n_i)}^{n} |\text{Solactual}| \leq \sum^{n} \text{Vecinos}(n_i) = m\right)$$

Solución alternativa:

Una idea fue Representar a G con listas de adyacencias.

Luego el algoritmo lo que haría es calcular el complemento del mismo.

Luego ejecutar un algoritmo de coloreo común donde simplemente se itera por todos los nodos, mirando a sus vecinos, y si hay alguna color entre 1 y |VecinosDe(nodo)| libre pintarlo con el mismo, esto es $O(n+m)$ el algoritmo y la respuesta final sería el Conjunto independiente del Grafo complemento que representa al color 1.

Lo malo de esta idea es que calcular el complemento de un grafo es de costo n^2 y no mejoraríamos la complejidad final.

La correctitud de este algoritmo se debe a que un C.I. en el complemento de un grafo es grafo completo en el grafo. Y si un nodo se hubiese ~~podido agregar al CI del color 1~~ podido agregar a la solución, este se hubiese podido agregar al conjunto independiente del color 1. Y como el algoritmo otorga como color de un nodo el menor posible, así hubiese sido.